RACE DETECTION IN MOS CIRCUITS BY TERNARY SIMULATION

Randal E. Bryant

Computer Science
California Institute of Technology

5091:TR:83

RACE DETECTION IN MOS CIRCUITS BY TERNARY SIMULATION

Randal E. Bryant

Computer Science
California Institute of Technology
Pasadena, CA 91125

5091:TR:83

To be presented at VLSI'83
Trondheim, Norway
August 16, 1983

This research was funded in part by the
Defense Advanced Research Contracts Agency
ARPA Order Number 3771
and by the
Caltech Silicon Structures Project

RACE DETECTION IN MOS CIRCUITS BY TERNARY SIMULATION

Randal E. Bryant

Computer Science
California Institute of Technology
Pasadena, CA 91125

Three-valued logic simulation, in which the third state X represents a signal in transition, has long been used for detecting races in logic gate circuits. This technique can also be applied to MOS digital systems represented at the switch level. In this paper we will describe ternary simulation, show examples of its use, and discuss its role in MOS design.

1. INTRODUCTION

Ternary simulation has long been used to detect potential timing errors in logic gate circuits.[6, 8, 1, 5] In this method, a signal in transition from 0 to 1 or from 1 to 0 is indicated by a third logic state X (sometimes denoted u or 1/2). To simulate a change in clock or data inputs the changing inputs are first set to X and the network is simulated according to the ternary extensions of the gate functions until a stable state is reached. As a consequence, all nodes which could possibly change state as a result of the input change are set to X. Then the changing inputs are set to their final values and the network is again simulated until a stable state is reached. As a result, any node for which the final logic level depends on the particular logic or wiring delays in the circuit will remain set to X, indicating a possible sequential timing error. Furthermore, a potential glitch (hazard) on a node is indicated by a state sequence of the form 0-X-0 or 1-X-1.

Such a simulation provides information that even the most accurate circuit simulator cannot -- whether the system will operate correctly (for the particular input sequences simulated) regardless of the actual delays in the circuit. Ternary simulation tests adherence to a timing discipline we shall call delay-insensitive design in which the sequential behavior of the system does not rely on the relative delays through the logic elements or wires as long as the controlling clocks operate slowly enough for the circuit to stabilize after each clock transition. Furthermore, ternary simulation requires only slightly more computational effort than ordinary logic simulation and hence can be used to check

large digital systems operating over long input sequences.

With the notable exception of level sensitive design [7], however, few logic designers using conventional technologies such as TTL adhere to such a strict timing discipline. Systems containing such components as edge-triggered flip-flops, one-shots, and carefully matched signal paths will fail ternary simulation, with most nodes ending up in the X state. No method has been developed to verify these systems will operate correctly over their entire range of manufacturing conditions and operating environments. Designers of integrated circuits in MOS technology, on the other hand, often follow timing conventions leading to delay insensitive designs for the entire system or at least for large sections of the system. Such conventions have been adopted due to the particular characteristics of integrated circuit manufacturing in which variations in the fabrication process create changes in the circuit parameters, circuits cannot be fine tuned after manufacture, and the exact time behavior of very large circuits cannot be simulated or analyzed with existing programs. For example, careful use of two or four phase nonoverlapping clocks [9] yields a delay-insensitive system as long as each of the clocks can be controlled separately. Even some of the more "exotic" circuit techniques such as dynamic memory, gated clocks, domino logic, and precharged busses can be used in delay insensitive systems. Of course, the use of dynamic memory or precharging also places a lower bound on the clocking rate, but this constraint is seldom hard to satisfy. By constructing a VLSI system out of delay-insensitive subsystems, the fine tuning of the circuits becomes largely a matter of performance optimization rather than a requirement for correct functionality. Thus the timing methodology required for a system to pass ternary simulation also leads to more reliable MOS systems.

The switch-level model was developed to describe the logical behavior of MOS digital systems.[2, 4] In this model, a circuit is represented as a network of transistor "switches". In keeping with the concept of a logic model, node voltages are represented by discrete states 0, 1, and X, and the electrical behavior is modeled in a highly idealized way. Ternary simulation can be applied to MOS digital systems modeled at the switch level as long as the behavior of the network is simulated properly when some of the nodes are in the X state, where "proper" will be defined shortly. The algorithm used by the simulator MOSSIM II[3] satisfies this requirement and can perform ternary simulation at a rate only 2 to 3 times slower than ordinary unit delay simulation. We have found ternary simulation a valuable tool in MOS design, often uncovering subtle, potentially fatal timing errors overlooked both by the designer and by ordinary logic simulation.

In this paper we will describe ternary simulation in the context of the

switch-level model.

2. THE SWITCH-LEVEL MODEL

The switch-level model consists of three parts: a network model for representing MOS circuits, a formal model describing the logical behavior of a network, and an algorithm for simulating the behavior. For this paper, we present only key characteristics of the model.

2.1. Network Model

A switch-level network consists of a set of nodes connected by a set of transistors. During a simulation, each node has state 0, 1, or X, where 0 and 1 represent low and high voltages, respectively, and X represents an unknown or invalid voltage. The X state can arise during ordinary simulation due to uninitialized nodes or invalid circuit operation (e.g. short circuits) and during ternary simulation due to a signal transition. Each transistor has a state 0, 1, or X, where 0 and 1 represent open (nonconducting) and closed (fully conducting) conditions respectively, and X represents an indeterminate condition between (inclusively) nonconducting and fully conducting.

Each node is classified as either an input node or a storage node. An input node provides a strong signal to the system, and its state is not affected by the actions of the network, much like a voltage source in an electrical network. Examples include the power and ground nodes Vdd and Gnd which act as constant sources of 1 and 0, respectively, as well as any clock or data inputs.

A storage node has state determined by the operation of the network, and it can hold this state in the absence of connections to input nodes, much like a capacitor in an electrical network. Each node can be assigned a size to indicate its approximate capacitance relative to other nodes with which it may share charge. For the purposes of this paper we will consider only 2 sizes: "small" and "large", although this can be generalized to more. Any time a set of storage nodes share charge without connections to input nodes, the outcome depends only on the initial states of the largest nodes in the set.

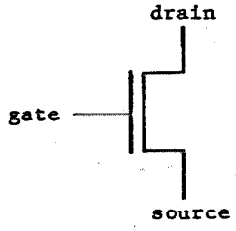

Figure 2-1: Transistor Diagram

A transistor is a three terminal device with terminals labeled "gate", "source", and "drain" as shown in Figure 2-1. No distinction is made between the

source and drain connections -- every transistor is a symmetric, bidirectional device. Each transistor has a type n, p, or d corresponding to n-type, p-type, and depletion mode devices, respectively. A transistor acts as a switch connecting its source and drain nodes controlled by the state of its gate node. The relation between gate state and transistor state depends on the transistor type as given by the following table:

gate state	n-type	p-type	d-type
0	0	1	1
1	1	0	1
X	X	X	1

Each transistor has a strength indicating its approximate conductance when closed relative to other transistors which may form part of a ratioed path. For the purpose of this paper we will consider only 2 transistor strengths: "weak" and "strong", although this can be generalized to more. When a storage node is connected to a set of input nodes by paths of conducting transistors, the outcome will depend only on the states of the input nodes connected by paths of maximum strength, where the strength of a path equals the minimum transistor strength in the path.

The switch-level network model attempts to capture those aspects of an MOS circuit which determine its logical behavior, while abstracting away the detailed electrical behavior. As a result of the abstraction, however, the model may not predict the true behavior of a circuit, especially in cases of marginal transistor or node sizing errors. Furthermore, the network model does not contain enough detail to accurately model the timing behavior, because even in circuits with straightforward logical behavior, the timing can be quite subtle.

2.2. The Steady State Response

The state of a switch-level network is given by three vectors x, y and z, where elements x_i, y_i, and z_i indicate the states of input node i, storage node i, and transistor i, respectively. In general, the state of a transistor is determined by the state of its gate node, and we will define the transistor state function Z(x,y) to yield the vector of transistor states created when the nodes are in states given by the vectors x and y.

The primitive behavior of a switch-level network is described by its steady state response which can be described informally as the set of states that would form on the storage nodes for a particular set of transistor states, input node states, and initial storage node states. That is, if we could hold the transistors fixed in states given by the vector z, set the input nodes to states given by the vector x, and initialize the storage nodes to states given by the

vector y, then the storage nodes would attain new states given by a vector y′ due to the conducting paths formed by the transistors to the storage nodes from input nodes and other storage nodes. Thus a network is characterized by its steady state response function Y where Y(x,y,z) = y′. The steady state response function is computed in MOSSIM II by solving a set of equations in a simple, discrete algebra to determine the effects of paths created by conducting transistors. These equations are solved by a relaxation algorithm with linear time complexity in the number of transistors.

2.3. Unit Delay Simulation

Switch-level simulators such MOSSIM II are designed primarily to simulate clocked systems in which the clocks change slowly enough for the circuit to stabilize after each input change. To simulate such a system, each clock cycle is divided into a sequence of phases where within each phase all inputs (both clock and data) are held fixed. The simulation of a single phase involves repeated computation of the steady state response function until a stable state is reached. More precisely, if the storage nodes are initially in state y and the input nodes are set to new state x′ the phase simulation proceeds as follows:

```
procedure PHASE(x′,y);
begin
    z := Z(x′,y);
    y′ := Y(x′,y,z);
    stepcount := 1;
    while y ≠ y′  &  stepcount ≤ limit do
    begin
        y := y′;
        z := Z(x′,y);
        y′ := Y(x′,y,z);
        stepcount := stepcount + 1;
    end
end
```

That is, the program repeatedly sets the transistors to states determined by the states of their gate nodes, computes the steady state response of the network, and sets the storage nodes to their steady state levels until either a stable state is reached or a maximum step limit is exceeded. By maintaining an event list of nodes which need to be updated, only portions of the network state need be computed in each step. This algorithm provides the user with a "unit delay" timing model in which a transistor switches one time unit (i.e. one computation of the steady state function) after its gate node changes state. The step limit is required to prevent unbounded oscillations caused by unstable circuits such as inverter rings and by anomalies resulting from perfectly matched unit delays. This timing model only loosely resembles the actual timing in the circuit, and hence unit delay simulation often fails to detect potential timing errors.

3. TERNARY LOGIC

The X state represents an uncertain or invalid node voltage or transistor conductance. Such states can arise during the normal operation of an MOS circuit due to (generally transient) short circuits or improper charge sharing. Furthermore, all nodes are set to X at the start of a simulation to indicate that they have not been properly initialized. Hence even for ordinary logic simulation the switch-level model must take account of the X state in a reasonable way, although accurate and efficient modeling is more crucial in ternary simulation. The use of a third or undefined logic value has been studied extensively for logic gate networks,[5, 1] and we will define its effect in switch-level networks in a similar way. In fact, switch-level simulations of logic gates implemented with conventional CMOS and nMOS circuit structures yield the same results as would a three-valued logic gate simulator.

We will refer to the set $T = \{0,1,X\}$ as the ternary domain and its subset $B = \{0,1\}$ as the Boolean domain. The elements of T are partially ordered $0 < X$ and $1 < X$ as denoted by the following Hasse diagram:

The least upper bound (l.u.b.) of a set of ternary values equals 1 (or 0) if and only if all elements of the set equal 1 (or 0), and equals X otherwise. Thus, the l.u.b. operation acts as a "consistency" operation with inconsistency represented by X.

For any function defined over a Boolean domain (but possibly having a ternary range), $f:B^n \rightarrow T$, its ternary extension is defined as the function $f^t:T^n \rightarrow T$ such that

$$f^t(a_1,\ldots,a_n) \doteq \text{l.u.b.}\{f(b_1,\ldots,b_n) \mid b_i \in B,\ b_i \leq a_i \text{ for all } 1 \leq i \leq n\}$$

That is, when some of the arguments to f^t equal X, the function yields a Boolean value if and only if it would have this same value with the arguments equal to X set to all possible combination of 0's and 1's. Thus, the ternary extension captures the concept that the value X represents an uncertain or ambiguous logic value which when given as an argument of a function will yield an uncertain or ambiguous result if and only if the output is sensitive to this input. This is precisely the property desired for ternary simulation in which X is used to indicate a signal in transition which can cause other transitions on nodes which are sensitive to this signal.

A function defined over a ternary domain is said to be ternary-consistent if it is equivalent to the ternary extension of some Boolean function. That is $g:T^n \rightarrow T$

is ternary consistent if for some $f:B^n \rightarrow T$, $g = f^t$. For example, the transistor state function Z is ternary-consistent for each transistor, because the state of a transistor with an X on its gate node equals 1.u.b(0,1) = X for both type n and type p transistors and equals 1.u.b(1,1) = 1 for type d transistors. It can also be shown that the steady state response function computed by MOSSIM II is ternary-consistent for each node. By simulating the system with a ternary-consistent steady state response function we strike a balance between excessive lenience, i.e. overlooking the effect of the uncertainty represented by an X, and excessive conservativism, i.e. spreading X's beyond the area of uncertainty.

4. TERNARY SIMULATION ALGORITHM

Ternary simulation is applied to logic gate networks by modeling the network according to the ternary extensions of the functions for each logic gate. We can achieve a similar effect in switch-level networks by modeling the network according to a ternary-consistent steady state response function. To simulate the effect of a change in input state from x to x' with the storage nodes initially in state y, we perform the following computation:

```
procedure TPHASE(x,x',y);
begin
    for i := 1 to m do t_i := 1.u.b.(x_i,x_i')
    ## Transition Phase ##
    z := Z(t,y);
    y' := Y(t,y,z);
    while y ≠ y' do
    begin
        y := y';
        z := Z(t,y);
        y' := Y(t,y,z)
    end;

    ## Stabilization Phase ##
    z := Z(x',y);
    y' := Y(x',y,z);
    while y ≠ y' do
    begin
        y := y';
        z := Z(x',y);
        y' := Y(x',y,z)
    end;
end
```

That is, every input i such that $x_i \neq x_i'$ is first set to X, and the network is simulated until a stable state is reached. This part of the simulation is termed the transition phase. Only state changes of the form 0-X and 1-X can occur, with X indicating a possible transition in node voltage. Then each input i is set to the new value x_i' and the network is again simulated until a stable state is reached. This part of the simulation is termed the stabilization phase. Only state changes of the form X-0 and X-1 can occur, with a state change indicating a

unique stable state for the node. For both phases, the computation is guaranteed to terminate as long as the initial state was itself stable, as can be shown by a simple monotonicity argument. Even the simulation of an inverter ring will terminate, but with all nodes in the ring set to X. An X on a node at the end of this computation indicates that the node does not have a stable final state as in an inverter ring, or that the final state of the node depends on the relative delays in the circuit elements and wires.

5. EXAMPLE

As an example of applying ternary simulation to a switch-level network, consider the network shown in Figure 5-1 consisting of a three-transistor dynamic RAM cell, and a bus with a static pullup and a pulldown controlled by the signal "down"

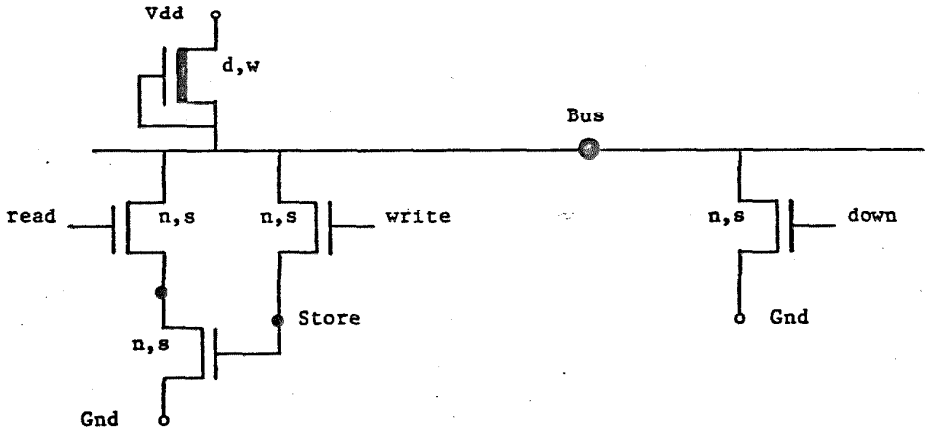

Figure 5-1: Switch-Level Model of Three Transistor Dynamic RAM

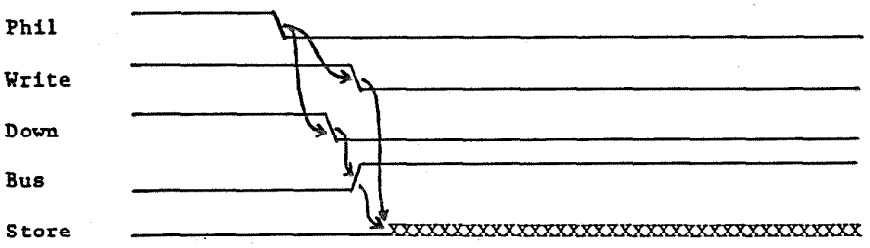

Figure 5-2: Timing Error When Both Write and Down are Synchronized by Phil

Suppose both the signals "Down" and "Write" are synchronized (i.e. And'ed) by the same clock Phil. Then as the clock turns off, a slight skew in the two signals could cause the pulldown to shut off, allowing the bus to rise, before the node "Store" has been isolated from the bus, thereby corrupting the stored value as shown in Figure 5-2. Thus, the circuit is delay-sensitive, depending on the

relative delays of the logic generating the control signals Write and Down. The sequence of network states in a ternary simulation of this circuit (and the control signal logic not shown in the illustration) would appear as follows:

	initial	transition	stabilization
Phi1	1	X X X	0 0 0
write	1	1 X X	X 0 0
down	1	1 X X	X 0 0
Bus	0	0 0 X	X X 1
Store	0	0 0 X	X X X

Thus, the timing error is successfully identified.

This error could be corrected in several different ways. First, we could simply make sure that the delay in the logic generating control signal "Down" is longer than the delay in the logic for "Write", perhaps using pairs of inverters as delay elements. This, however, would still give a delay-sensitive design and ternary simulation would yield the same result no matter how much delay is inserted.

Second, we could modify the logic for the bus pulldown as shown in Figure 5-3, so that the the bus is held low until the second (nonoverlapping) clock Phi2 goes high. This circuit relies on dynamic storage of charge on the node Hold while both clocks are low. The sequence of states in a ternary simulation of Phi1 going low and then Phi2 going high would appear as follows:

	initial	transition	stabilization	transition	stabilization
Phi1	1	X X	0 0	0 0	0 0
Phi2	0	0 0	0 0	X X	1 1
write	1	1 X	X 0	0 0	0 0
down	1	1 X	X 0	0 0	0 0
Bus	0	0 0	0 0	0 X	X 1
Store	0	0 0	0 0	0 0	0 0

This modification results in a delay-insensitive design, because the control signal for the bus pulldown is delayed by an external clock rather than by internal circuit delays.

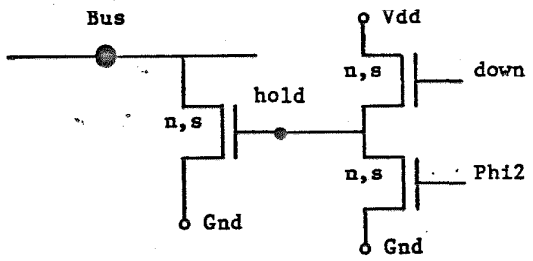

Figure 5-3: Alternate Implementation of Bus Pulldown

Finally we could change to a precharged bus, replacing the static pullup with

an n-type transistor which charges the bus to 1 every time clock Phi2 goes high as shown in Figure 5-4. This modification also gives a delay-insensitive design, because the bus will stay low even after the pulldown transistor shuts off until the following Phi2 phase. We rely on the large capacitance of the bus to overwrite the previous value on node Store during a memory write. A ternary simulation of this circuit would give the same states shown for the previous case.

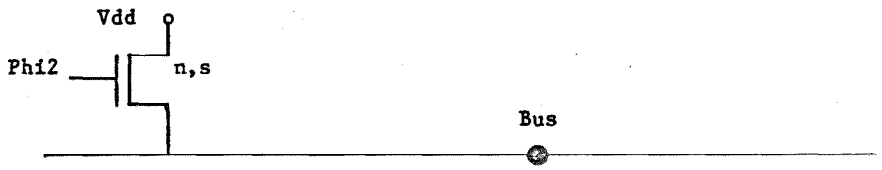

Figure 5-4: Alternate Implementation of Bus Pullup

6. CONCLUSION

We have applied ternary simulation to a variety of circuits designed in both nMOS and CMOS technologies using a variety of different clocking schemes and circuit design styles. It has proved a valuable tool for helping the designer identify delay sensitivities which could then either be eliminated by modifying the circuit or be analyzed more carefully by circuit simulation.

As a further extension of ternary simulation, the range of application would be increased if the user were allowed to specify delay sensitivities which were known to work correctly. For example, suppose in the network of Figure 5-1 the user could specify the control signal Write will always reach 0 before the control signal Down starts to fall.[1] The simulator would then take such constraints into account, rather than always assuming worst case behavior. Such an extension is required before ternary simulation can be applied to self-timed systems, because these circuits rely heavily on the relative logic and wiring delays in the sequencing elements. Unfortunately, no general algorithm has been devised for employing this constraint information.

The concept of having a simulator test for adherence to a particular design discipline seems to have great potential, especially for disciplines which can be tested by well-defined algorithms. Whereas the level of abstraction provided by logic simulation is usually viewed as a necessary, but not particularly desirable trade-off between accuracy and computational efficiency, ternary simulation exploits the power of this abstraction to help the designer create more reliable systems. We have implemented several other such tests in MOSSIM II, including

---

[1] Note how constraints are specified in terms of the relation between the completion of one signal transition and the start of another. Ternary simulation does not assume transitions occur instantaneously!

tests for threshold drops or rises in CMOS, unrefreshed stored charge, and a class of glitches caused by transient charge sharing effects. In all cases, the X state is used to indicate a corrupted node state, and the effects of these corrupted states is modeled by simulating the X's according to the ternary-consistent steady state response function. Since all of these tests are applied by setting mode switches in the simulator, they do not require reformatting the network description or the test vectors.

7. ACKNOWLEDGMENTS

Mike Schuster has provided invaluable assistance in implementing the simulator MOSSIM II and in furthering the development of the switch-level model. This research was funded in part by the Defense Advanced Research Contracts Agency ARPA Order Number 3771 and by the Caltech Silicon Structures Project.

[1]
Breuer, M. A.
A Note on Three-Valued Logic Simulation.
IEEE Transactions on Computers C-21(4):399-402, April, 1972.

[2]
Bryant, R.
A Switch-Level Model of MOS Logic Circuits.
In J. Gray, editor, VLSI 81, pages 329-340. Academic Press, August, 1981.

[3]
Bryant, R., Schuster, M., and Whiting, D.,
MOSSIM II: A Switch-Level Simulator for MOS LSI, User's Manual
Department of Computer Science, California Institute of Technology, 1982.

[4]
Bryant, R. E.
A Switch-Level Model and Simulator for MOS Digital Systems.
Technical Report 5065, Department of Computer Science, California Institute of Technology, January, 1983.

[5]
Brzozowski, J. A., and M. Yoeli.
On a Ternary Model of Gate Networks.
IEEE Transactions on Computers C-28(3):178-183, March, 1979.

[6]
Eichelberger, E. B.
Hazard Detection in Combinational and Sequential Switching Circuits.
IBM Journal of Research and Development 8:90-99, 1965.

[7]
Eichelberger, E. B., and T. W. Williams.
A Logic Design Structure for LSI Testability.
Journal of Design Automation and Fault-Tolerant Computing 2(2):165-178, May, 1982.

[8]
Jephson, J. S., R. P. McQuarrie, and R. E. Vogelsberg.
A Three-Level Design Verification System.
IBM Systems Journal 8(3):178-188, March, 1969.

[9]
Mead, C., and Conway, L.
Introduction to VLSI Systems.
Addison Wesley, 1980.